The Yale Undergraduate Research Journal

Volume 1
Issue 1 *Fall 2020*

Article 12

2020

Applications of Bayesian Inference for Modelling Dynamic Instability in Neuronal Dendrite Morphogenesis

Daniel Fridman
Yale University

Applications of Bayesian Inference for Modelling Dynamic Instability in Neuronal Dendrite Morphogenesis

Cover Page Footnote

I would like to thank Prof. Jonathon Howard, Olivier Trottier, Sabya Sutradhar and the rest of the Howard lab for their support and feedback throughout this project.

STEM I Statistics VOL. 1.1 I Oct. 2020

Applications of Bayesian Inference for Modelling Dynamic Instability in Neuronal Dendrite Morphogenesis

Daniel Fridman[1]

[1]*Yale University*

Abstract

Neurons are complex biological systems which develop intricate morphologies and whose dendrites are essential in receiving and integrating input signals from neighboring neurons. While much research has been done on the role of dendrites in neuronal development, a further understanding of dendrite dynamics can provide insight into neural development and the cellular basis of neurological diseases such as schizophrenia, Down's syndrome, and autism. The Jonathon Howard lab hypothesizes that microtubules are a primary driving force in dendrite dynamics. Since it is known that microtubules display dynamic instability, rapidly transitioning between growth, paused, and shrinking states, the Howard lab proposes a similar 3-state transition model for dendrite dynamics. However, this model remains to be rigorously evaluated on dendrite branch data. In this paper, I develop a novel implementation of the Gibbs sampling algorithm for parameterization of the proposed 3-state mixture model, improving upon prior parameterization methods such as least squares fitting. Furthermore, I apply the algorithm on a confocal microscopy dataset of measured dendrite branch velocities from Class IV dendritic arbors in *Drosophila melanogaster*, demonstrating a good fit of the model to the data.

1. INTRODUCTION

1.1 Neuronal Dendrite Morphogenesis

Neurons are extraordinarily complex biological systems whose morphological structure and dynamics allow them to efficiently process signals and form the circuitry of the brain. Dendrites, which branch out of the neuron's cell body, play a crucial role in receiving and integrating input signals from neighboring neurons. A neuron's specific dendrite morphology and patterning plays an important role in determining which signals the neuron receives and how it processes them. Understanding dendrite morphology and dynamics as well as the underlying mechanisms driving dendritic development has important implications for elucidating neural and brain development as well as enhancing our understanding of the cellular basis of neurological and neurodevelopmental disorders.

Over the past several decades, studies on *Drosophila melanogaster* neurons have revealed a broad range of genetic, molecular, and biophysical mechanisms contributing to dendrite morphogenesis (1). In particular, it has been shown that microtubules play essential roles in dendrite growth, dynamics, and patterning (1). As a result of these mechanisms, different neurons develop distinct dendrite morphologies including different dendrite sizes, branching patterns, and area coverage (dendritic field). These structural differences allow certain neurons to carry out distinct physiological functions within the neural circuitry of the brain. In particular, four distinct classes

of dendritic arborization neurons have been identified in *D. melanogaster* (1).

1.2 Modelling Dendrite Branch Dynamics

Since microtubules play important roles in dendrite dynamics (1), the Jonathon Howard lab hypothesizes that dendrites should display similar dynamic properties to microtubules. In particular, it is known that microtubules display dynamic instability, rapidly transitioning between growing, shrinking, and paused states on the order of minutes (2). Such rapid transitions allow microtubules to efficiently adopt new spatial arrangements in response to cellular needs and changes in the environment (2). It stands to reason that dendrites would take advantage of microtubule dynamic instability for dendrite branch development, attainment of particular dendrite morphologies and branching patterns, and rapid response to stimuli from neighboring neurons. The Howard lab thus hypothesizes that dendrite branches should display the same three dynamic branching states – growing, paused, and shrinking – as can be observed in microtubules.

Studies in the Howard lab have focused on dendrite dynamics and branching processes in Class IV dendritic arborization neurons of *D. melanogaster*. Using confocal microscopy, the Howard lab tracked the spatial and temporal dynamics of dendrite branch tips, recording a time series of branch lengths. Each time series consisted of a track of a single dendrite branch length for 30 minutes with 441 total tracks recorded. From this data, the corresponding dendrite branch velocities were computed. A histogram of the raw velocity data is shown below (Fig. 1).

Building upon the 3-state hypothesis for dendrite dynamics, the Howard lab hypothesizes that dendrite branch velocities from Class IV dendrites in *D. melanogaster* can be segmented into distinct growing, paused, and shrinking state velocities. Furthermore, the velocities of each state can be represented according to a unique velocity distribution which can be modelled as a Gaussian for the paused state, log-Normal for the growing state, and negative log-Normal for the shrinking state. As such, the dendrite branch velocity data can be modelled as a three-state log-N-Gauss-log-N mixture model with unique mean, variance, and weight parameters (Eq. 1) where y^+ refers to only positive velocity values in the dataset (for the log-Normal growth state) and y^- refers to only negative velocity values (for the negative log-Normal shrinking state).

Equation 1

$$y_i \sim w_1 \frac{1}{(y^+)\sigma_1\sqrt{2\pi}} exp(-\frac{(ln(y^+)-\mu_1)^2}{2\sigma_1^2}) + w_2 \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(y-\mu_2)^2}{2\sigma_2^2}) + w_3 \frac{1}{|y^-|\sigma_3\sqrt{2\pi}} exp(-\frac{(ln(|y^-|)-\mu_3)^2}{2\sigma_3^2})$$

1.3 Applying Bayesian Inference for Model Parameterization

In recent years, Bayesian inference has gained popularity for model parameterization. Through the application of Bayes rule, Bayesian inference allows for calculating posterior distributions for model parameters that can be updated with new data. Furthermore, in cases where models are too complex to analytically calculate posterior distributions, Markov Chain Monte Carlo (MCMC) methods have allowed for estimating posterior distributions by iteratively sampling from them. One such MCMC method is Gibbs sampling, which will be discussed in detail below. In this paper, I develop a novel implementation of the Gibbs sampling algorithm in order to parameterize the proposed log-N-Gauss-log-N mixture model for class IV dendritic arbors using Gibbs sampling. Furthermore, using Gibbs sampling, I seek to develop a statistically rigorous method for segmenting dendrite branch data into the hypothesized growing, paused, or shrinking dynamic states. The results of this model parameterization will allow for the assessment of the 3-state hypothesis for dendritic

development, providing further insight into the dynamics of dendrite morphogenesis.

2. BACKGROUND ON GIBBS SAMPLING

In this section I will introduce the generalized Gibbs sampling algorithm and its application towards Gaussian models, leading up to my specified implementation of the Gibbs sampling algorithm for paramaterizing a log-N-Gauss-log-N mixture model.

2.1 Bayesian Inference

In many diverse fields, scientists often use statistical models to explain and better understand complex, noisy natural processes. The goal of such modelling is to derive a model that adequately explains experimentally measurable or observable data. In order to do so, researchers are often faced with the task of estimating model parameters from the data. This task is known as statistical inference (3).

While traditionally, least-squares fitting methods as well as frequentist-based inference and maximum likelihood estimation (MLE) have been used to estimate model parameters, they are only capable of providing point estimates of parameter values. On the other hand, Bayesian inference provides a rigorous method for determining posterior probability distributions of the parameter space. The basis of Bayesian inference is the Bayes' rule. If we have a hypothesized model with parameters θ and observed or measured data D, we are able to use Bayes' rule to make the following inversion: $p(D|\theta) \rightarrow p(\theta|D)$, using the following equation (3):

Equation 2

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

where $p(D|\theta)$ is known as the *likelihood*, $p(\theta)$ is known as the *prior*, and $p(\theta|D)$ is known as the *posterior*. The *likelihood* represents the probability of generating a certain sample of data D, given that we know the model that generated our data and that our model's parameters equal θ. The *prior* represents an initial assumption about the distribution of our parameter space before seeing the data. The denominator on the right-hand side is known as the *marginal likelihood* and represents the probability of obtaining a certain set of data, assuming we have a defined likelihood and a prior. Finally, and most importantly, the *posterior* is the end goal of Bayesian inference and represents our updated distribution across the parameter space after seeing the data (3).

2.2 Gibbs Sampling Overview

While closed-form solutions of the posterior distributions for simple models can be obtained using Bayesian inference, more complex models with many parameters may have no such solutions. Thus, it may not be possible to obtain exact posteriors for the parameters of complex models. Nonetheless, posteriors can be estimated using dependent sampling methods referred to as Markov Chain Monte Carlo (MCMC). The idea of MCMC sampling is that the posterior can be sampled from and given enough samples, an approximation to the true posterior can be obtained.

One type of MCMC algorithm is known as *Gibbs sampling*. The idea of Gibbs sampling is that while it may not be possible to obtain a closed-form solution for the multi-parameter posterior, it may be possible to obtain closed-form posteriors for single model parameters conditioned on the other parameters (using the idea of *conjugate priors* (4,5,6), Appendix A). Thus, each parameter can be sampled from individually, dependent on the other parameters and the data. Sampling for multiple iterations and updating the parameter values across every iteration, the posterior for each parameter can be recreated (essentially returning a cross-section of each parameter dimension in the original multi-dimensional posterior) (3).

2.2.1 Generalized Gibbs Sampling Algorithm

As a generalized example of the Gibbs sampling procedure, we can imagine that we have a model with N unknown parameters, $\Theta = (\theta_1, \theta_2, \dots, \theta_N)$ associated with a model that we've hypothesized for our data. We also assume that we have an observed dataset, D. Our goal is to estimate the N-dimensional posterior, $p(\theta_1, \theta_2, \dots, \theta_N|D)$. While we may be unable to obtain a closed-form solution for this posterior, we may instead be able to obtain closed-form solutions for the conditional posteriors of each of the parameters individually: $p(\theta_1|\theta_2, \dots, \theta_N, D), p(\theta_2|\theta_1, \theta_3, \dots, \theta_N, D), \dots, p(\theta_N|\theta_1, \dots, \theta_{N-1}, D)$ We can then apply the Gibbs sampling algorithm to sample from each of the conditional posteriors and estimate the N-dimensional posterior according to Algorithm 1 below (6).

Algorithm 1 Generalized Gibbs Sampling Algorithm

Initialize at a random starting point $(\theta_1^0, \theta_2^0, \dots, \theta_N^0)$
for t in t_iterations **do**
  randomize the order of sampling from conditional posteriors
  sample each parameter using the most recently sampled parameter values, i.e. for an order of $(\theta_1, \theta_2, \dots, \theta_N)$, update as:
  $\theta_1^t \sim p(\theta_1^t|\theta_2^{t-1}, \dots, \theta_N^{t-1}, D)$
  $\theta_2^t \sim p(\theta_2^t|\theta_1^t, \theta_3^{t-1}, \dots, \theta_N^{t-1}, D)$
  ...
  $\theta_N^t \sim p(\theta_N^t|\theta_1^t, \dots, \theta_N^t, D)$
  append $(\theta_1^t, \dots, \theta_N^t)$ to $[\theta_1\_vals, \dots, \theta_N\_vals]$
end for
return $[\theta_1\_vals, \dots, \theta_N\_vals]$

2.2.2 Gibbs Sampling Example for Simple Gaussian Model

As a specific application of the Gibbs sampling procedure, we will look at a Gaussian model with two unknown parameters, μ and σ. Assuming that our data is generated from a Gaussian distribution, $y_i \sim \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(y-\mu)^2}{2\sigma^2})$, our model has a Gaussian likelihood for N samples. We seek to determine a 2-dimensional posterior, $p(\mu, \sigma|y)$. Using the idea of conjugate priors, we can determine the closed-form solutions for both the μ and σ parameters conditioned on the other parameter and our data as follows:

It has been shown that the following priors are conjugate to the Gaussian likelihood (7):

Equation 3

$$\mu|\tau \sim N(\mu_0, n_0\tau)$$
$$\tau \sim Gamma(\alpha, \beta)$$

where $\tau = \frac{1}{\sigma^2}$. The corresponding posteriors can then be derived from the priors above (7):

Equation 4

$$\tau|y \sim Gamma\left(\alpha + n/2, \beta + \frac{1}{2}\sum(y_i - \bar{y})^2 + \frac{nn_0}{2(n+n_0)}(\bar{y} - \mu_0)^2\right)$$
$$\mu|\tau, y \sim N\left(\frac{n\tau}{n\tau + n_0\tau}\bar{y} + \frac{n_0\tau}{n\tau + n_0\tau}\mu_0, n\tau + n_0\tau\right)$$

We can thus determine the posterior $p(\mu, \sigma|y)$ by using Gibbs sampling to iteratively sample from the μ and σ conditional posteriors, respectively, and updating our parameter values as described in algorithm 2 below:

Algorithm 2 Gibbs Sampling Algorithm for Simple Gaussian Model

Initialize at a random starting point (μ^0, σ^0)
for t in t_iterations **do**
  sample each parameter using the most recently sampled parameter values, update as:
  $\tau^t|y \sim \Gamma\left(\alpha + n/2, \beta + \frac{1}{2}\sum(y_i - \bar{y})^2 + \frac{nn_0}{2(n+n_0)}(\bar{y} - \mu_0)^2\right)$
  $\mu^t|\tau^t, y \sim N\left(\frac{n\tau^t}{n\tau^t + n_0\tau^t}\bar{y} + \frac{n_0\tau^t}{n\tau^t + n_0\tau^t}\mu_0, n\tau^t + n_0\tau^t\right)$
  append $(\mu^t, \sqrt{\frac{1}{\tau^t}})$ to $[\mu\_vals, \sigma\_vals]$
end for
return $[\mu\_vals, \sigma\_vals]$

3. IMPLEMENTATION AND APPLICATION TO DENDRITE MORPHOGENESIS

In this section I will describe the implementation of the Gibbs sampling algorithm for the 3-component log-N-Gauss-log-N mixture model used to model dendrite branch velocity distributions. I will first discuss the methods employed in applying Gibbs sampling to mixture models and then discuss the specifics of my implementation.

3.1 Gibbs Sampling for Mixture Models (6)

Mixture models contain multiple component distributions and thus require parameters to be sampled for each component in order to estimate the posterior. In order to accomplish this, a trick known as *data augmentation* is used which adds a new latent indicator variable to the data to label which component each data point was likely drawn from. For a k-component mixture model, we would have k potential categories for each indicator variable: $cat_i \in (1,2,\dots,k)$. Additionally, we assume that in total our mixture model contains (D+k) parameters representing D parameters from all the components of the model and k weight parameters associated with each of the k components. With the inclusion of latent variables, the posterior (originally with D+k parameters) now contains N additional parameters indicating the category of each data point: $p(\theta_1,\dots,\theta_D,w_1,\dots,w_k,cat_1,\dots,cat_N|y)$. These latent variables will be marginalized out in the process of Gibbs sampling, but are included to simplify the sampling procedure.

After including the latent indicator variables, the following conditional posteriors need to be computed in order to apply the Gibbs sampling procedure:

Equation 5

$$p(\theta_1|\dots) \propto p(y|\dots)p(\theta_1),\dots,p(\theta_N|\dots) \propto p(y|\dots)p(\theta_N)$$
$$p(w_1|\dots) \propto p(y|\dots)p(w_1),\dots,p(w_k|\dots) \propto p(y|\dots)p(w_k)$$
$$p(cat_1|\dots) \propto p(y|\dots)p(cat_1),\dots,p(cat_N|\dots) \propto p(y|\dots)p(cat_N)$$

The way this can be achieved is by using the idea of conjugate priors (Appendix A) to find an appropriate prior to each of the likelihoods and thus obtain a closed-form conditional posterior for each parameter. Then, the conditional posteriors for each of the parameters can be sampled from and updated iteratively.

The posterior $p(\theta_i|\dots)$ can be computed using the conjugate prior to the likelihood of whichever distribution our k-th component of the model comes from. For example, if one of our model components comes from an exponential distribution, we would use a Gamma prior and its corresponding posterior as shown in Appendix A. Likewise, if one of our model components comes from a Gaussian distribution, we would use a $N - \Gamma^{-1}$ prior and its corresponding posterior as shown in section 2.2.1. The posterior for the k-th component, however, would be conditioned on the data assigned to the k-th component rather than the full dataset.

Next, in order to assign each data point to one of k components, we need to sample cat_i from k components with probability equal to the posterior probability of cat_i coming from each of k components, $p(cat_i = 1|\dots),\dots,p(cat_i = k|\dots)$. This posterior probability can be expressed as follows:

Equation 6

$$p(cat_i = j|\dots) \propto p(y_i|cat_i = j,\dots)p(cat_i = j)$$
$$\propto p(y_i|cat_i = j,\dots) * w_j$$

As shown above, the posterior probability that data point i is assigned to category j is proportional to the likelihood of data point i being drawn from the j-th model component times the weight of the j-th component.

Each data point in the dataset is then assigned to one of k possible categories according to a categorical distribution with corresponding probabilities:

Equation 7

$$cat_i \sim Categorical(cat_i|p_1,\dots,p_k)$$

where $p_1 = p(cat_i = 1|\dots),\dots,p_k = p(cat_i = k|\dots)$. The categorical distribution is an extension of the Bernoulli distribution to k dimensions and can be thought of as doing a k-dimensional coin flip with corresponding probabilities as the weights of each side of the k-dimensional coin.

The final parameters for which we need to determine a conditional posterior are the weight parameters w for each of the k model components. It's important to realize that the weight w_j essentially represents the probability of sampling from the j-th component and thus (in order to ensure a valid probability distribution) the weights in the mixture model need to sum to 1, $w_1 + w_2 + \cdots + w_k = 1$.

Using the conjugacy between a categorical likelihood and the Dirichlet prior, we can obtain a closed form for the joint posterior for all k weight parameters as follows:

Equation 8

$$\begin{aligned} p(w_1, \dots, w_k| \dots) &\propto L(cat_i| \dots)p(w_1, \dots, w_k) \\ &\propto L(cat_i| \dots) * Dir(w_1, \dots, w_k|\alpha_1, \dots, \alpha_k) \\ &= L(cat_i| \dots) * \frac{\Gamma(\sum_{j=1}^{k} \alpha_j)}{\prod_{j=1}^{k} \Gamma(\alpha_j)} \prod_{j=1}^{k} w_j^{\alpha_j - 1} \\ &\propto Dir(w_1, \dots, w_k|n(cat_1) + \alpha_1, \dots, n(cat_k) + \alpha_k) \end{aligned}$$

where $n(cat_j)$ represents the number of elements assigned to category j.

With the steps above, we have derived the conditional posteriors for all of our model parameters and can now apply the Gibbs sampling algorithm to estimate the posterior of any mixture model whose likelihoods of its individual components have conjugate priors (i.e. for which $p(\theta_i| \dots)$ can be solved).

In the following section we will apply the steps shown in section 3.1 as well as the posterior for a Gaussian likelihood stated in section 2.2.2 to implement the Gibbs sampling algorithm for a 3-component log-N-Gauss-log-N mixture model.

3.2 Gibbs Sampling for 3-component log-N-Gaussian-log-N Mixture Model

As stated in section 1.2, we hypothesize that dendrite branches display growing, paused, and shrinking states. As a result, dendrite branch velocity data can modelled as being distributed according to a 3-component log-N-Gaussian-log-N mixture model containing 9 mean, variance, and weight parameters that we seek to determine (Eq. 1) (i.e. $\mu_{growing}$, μ_{paused}, $\mu_{shrinking}$; $\sigma_{growing}$, σ_{paused}, $\sigma_{shrinking}$; $w_{growing}$, w_{paused}, $w_{shrinking}$).

3.2.1 Deriving Conditional Posterior Distributions

In this section I will derive the conditional posterior parameter distributions for the μ and σ parameters of the 3-component log-N-Gaussian-log-N mixture model.

It is first important to note that any data distributed according to a log-Normal or negative log-Normal distribution can be transformed into a Gaussian distribution through a log transformation:

Equation 9

$$\begin{aligned} y \sim \text{log-Normal}(\mu, \sigma^2) &\rightarrow ln(y) \sim \mathcal{N}(\mu, \sigma^2) \\ y \sim \text{Negative log-Normal}(\mu, \sigma^2) &\rightarrow ln(|y|) \sim \mathcal{N}(\mu, \sigma^2) \end{aligned}$$

Then, assuming the data is either generated from a Gaussian distribution or can be transformed to follow a Gaussian distribution with parameters μ and τ, a Gaussian likelihood can be used for each component of the mixture model as follows:

Equation 10

$$\begin{aligned} y_{paused}|\mu_{paused}, \tau_{pasued} &\sim N(\mu_{paused}, \tau_{paused}) \\ ln(y_{growing})|\mu_{growing}, \tau_{growing} &\sim N(\mu_{growing}, \tau_{growing}) \\ ln(|y_{shrinking}|)|\mu_{shrinking}, \tau_{shrinking} &\sim N(\mu_{shrinking}, \tau_{shrinking}) \end{aligned}$$

where $\tau = \frac{1}{\sigma^2}$.

Given a Gaussian distributed dataset for each model component with unknown parameters μ and σ and their corresponding conditional posteriors (Eq. 9), the $N\text{-}\Gamma^{-1}$ distribution can be sampled from to generate an approximation of the posterior parameter distributions according to algorithm 3:

Algorithm 3 Sampling from the N-Γ^{-1} posterior

Initialization:
$\mu_0 = mean(\texttt{data})$
$n_0 = 2$
$\alpha = length(\texttt{data})$
$\beta = \frac{1}{2}\sum(\texttt{data} - \mu_0)^2$
$A = length(\texttt{data})$

for i in n_samples **do**
  $\alpha' \leftarrow \alpha + A/2$
  $\beta' \leftarrow \beta + \frac{1}{2}\sum(\texttt{data}_i - mean(\texttt{data}))^2 + \frac{A*n_0}{2(A+n_0)}(mean(\texttt{data}) - \mu_0)^2$
  $\tau|\texttt{data} \sim \Gamma(\alpha', \beta')$
  $\sigma^2_{posterior} \leftarrow \frac{1}{\tau}$
  append $\sigma^2_{posterior}$ to σ_vals list
  $\mu \leftarrow \frac{A}{A+n_0}\bar{x} + \frac{n_0}{A+n_0}\mu_0$
  $\sigma^2 \leftarrow \frac{1}{A\tau+n_0\tau}$
  $\mu_{posterior}|\tau, \texttt{data} \sim \mathcal{N}(\mu, \sigma^2)$
  append $\mu_{posterior}$ to μ_vals list
end for
return μ_vals, σ_vals

3.2.2 Defining the Gibbs Sampling Algorithm

The Gibbs sampling algorithm can be defined according to algorithms 3 and 4.

Algorithm 4 Gibbs Sampling for log-N-Gauss-log-N Mixture Model

for iteration t = 1, 2, . . .,N **do**
  if t=1 **then**
    Use Otsu initialization, . . .
    Compute k1 and k2 thresholds to threshold 3 categories
    Growing_Set $\leftarrow data > k2$
    Shrinking_Set $\leftarrow data < k1$
    Paused_Set $\leftarrow k1 < data < k2$
  else
    Compute probabilities for growing, shrinking, or paused categories based on μ, σ, and w parameters sampled at iteration $t-1$
    $p1 \leftarrow w^{t-1}_{growing} * log\text{-}N(\texttt{data\_set} > 0, \mu^{t-1}_{growing}, \sigma^{t-1}_{growing})$
    $p2 \leftarrow w^{t-1}_{paused} * N(\texttt{data\_set}, \mu^{t-1}_{paused}, \sigma^{t-1}_{paused})$
    $p3 \leftarrow w^{t-1}_{shrinking} * log\text{-}N(abs(\texttt{data\_set} < 0), \mu^{t-1}_{shrinking}, \sigma^{t-1}_{shrinking})$
    *Fill p1 and p3 with zero's for $p(\texttt{data\_set} < 0)$ and $p(\texttt{data\_set} > 0)$, respectively
    Assign each point in data_set as coming from growing, paused, or shrinking category with prob. p1,p2,p3, respectively
    for i in length(data_set) **do**
      $cat_i \sim Categorical(p1[i], p2[i], p3[i])$
    end for
    Reassign categories based on categorical samples, . . .
    Growing_Set $\leftarrow data[cat_i == 1]$
    Paused_Set $\leftarrow data[cat_i == 2]$
    Shrinking_Set $\leftarrow data[cat_i == 3]$
  end if
  Sample μ and σ parameter values according to normal-inverse-gamma posterior (*See Algorithm 3)
  $[\mu^t_{grow}, \sigma^t_{grow}] \sim N\text{-}\Gamma^{-1}(log(\texttt{Growing\_Set}))$
  $[\mu^t_{pause}, \sigma^t_{pause}] \sim N\text{-}\Gamma^{-1}(\texttt{Paused\_Set})$
  $[\mu^t_{shrink}, \sigma^t_{shrink}] \sim N\text{-}\Gamma^{-1}(log(abs(\texttt{Shrinking\_Set})))$
  Sample weight w parameter values according to Dirichlet posterior
  $[w^t_{grow}, w^t_{pause}, w^t_{shrink}] \sim Dirichlet(len(\texttt{Growing\_Set}) + \alpha_1, len(\texttt{Paused\_Set}) + \alpha_2, len(\texttt{Shrinking\_Set}) + \alpha_3)$
  *Repeat lines 8-28 for t = 2,...,N
end for

4. RESULTS

In order to paramaterize the dataset of dendrite branch velocities (Fig. 1) using the 3-component log-N-Gauss-log-N mixture model (Eq. 1), the Gibbs sampling algorithm (Algorithms 3,4) was applied on both simulated and real datasets and the results are described below.

4.1 Effects of Gibbs Sampling Initialization on Posterior Predictions

In order to verify that the Gibbs sampling algorithm successfully converged to the true posteriors, we tested the algorithm's performance on a simulated dataset with known parameter values. A dataset was simulated according to the 3-component log-N-Gauss-log-N mixture model with true μ, σ, and w parameters set to parameter values previously determined by the Howard lab for the dendrite branch velocity dataset using least-squares fitting (Fig. 3, Table 1).

The Gibbs sampling algorithm was initialized with random initialization, assigning each data point in the dataset to a growing, shrinking, or paused state with equal probability, with the restriction that only positive values could be assigned to a growing state and only negative values could be assigned to a shrinking state. Additionally, since it is known that the mean velocity of the paused state is 0 $\mu m/min$, the μ_{paused} parameter was fixed to 0. As shown in Figure 4 below, Gibbs sampling with random initialization failed to accurately recover the true parameters. Note that only the σ posteriors are shown, but the algorithm failed to recover μ and w posteriors as well.

Upon examination of the fitted distribution using the parameter means of the posteriors recovered by Gibbs sampling (Fig. 5), it is apparent that random initialization assigns many

large negative and large positive values to the Gaussian paused state, causing difficulties for the algorithm to converge and causing it to falsely converge to a wide Gaussian (large σ_{paused} (not shown)). Additonally, the algorithm converges to mean weights of about 0.91 for the Gaussian paused state and only about 0.046 and 0.042 for the log-Normal growing and shrinking states, respectively (posteriors not shown). Thus, it can be concluded that random initialization causes the algorithm to fit a wide Gaussian around the entire dataset, mostly disregarding the other components of the mixture model. This failure to converge to the true posterior may be attributed to the issue of multimodality in which the posterior contains multiple 'modes' of high probability parameter values and initialization far from the 'true mode' causes our sampler to converge to a lower probability mode.

To address the issue of multimodality, it stands to reason that initializing the sampler closer to the true posterior mode would facilitate proper convergence. In order to accomplish this, initializing the data segmentation from the mixture model into proposed growing, shrinking, and paused datasets such that the segmentation is closer to the true growing, shrinking, and paused datasets would aid in proper convergence of the sampler. Thus, a technique called Otsu's method was employed to better initialize the categories of the data. Otsu's method is a technique used in image processing for image thresholding. The idea of Otsu's method is to maximize the inter-class variance between any multi-class dataset (8). In our case, Otsu's method was implemented to threshold our dataset into 3 categories which were used to initialize the proposed data segmentation in the Gibbs sampler (Algorithm 4, lines 2-7) (Fig. 6).

Running the Gibbs sampling algorithm for 1000 iterations using Otsu's initialization successfully recovered the true parameters within 95% confidence intervals as shown in Figures 7-9.

Taking the mean of each of the parameter's posterior estimates from Gibbs sampling and plotting the fitted distribution overlayed with the true distribution shows that Gibbs sampling with Otsu initialization is successfully able to recover the true distribution and its parameters (Fig. 10, Table 1). In order to further assess the fit of the estimated distribution to the true distribution, the Kullback-Leibler (KL) divergence (10) was computed to be 0.0195, indicating an extremely good fit.

4.2 Parameterization of Experimentally Obtained Dendrite Branch Velocity Distribution

After successfully recovering the true parameters for the simulated model, I returned to my original goal of paramaterizing the experimental dataset of neuronal dendrite branch velocities (Fig. 1). As stated previously, the Howard lab hypothesizes that dendrite branch velocity distributions follow a log-N-Gauss-log-N model with distinguishable growing, paused, and shrinking state velocities. The Gibbs sampling algorithm (Algorithm 4) with Otsu initialization can then applied to the experimentally measured dataset after fixing μ_{paused} to 0. In order to increase confidence in posterior convergence, multiple MCMC chains were run. The posterior estimates for 5 MCMC chains with 95% confidence intervals are shown in Figures 11-13 and Table 2. In order to assess convergence of the Gibbs sampler, the Gelman-Rubin convergence diagnostic (3,9) was used and produced r_hat values below the threshold of 1.1, indicating that the MCMC chains had converged for all parameters (as shown in figures 11-13). Additionally, the effective sample size (3) was computed for 1000 MCMC iterations (700 iterations after convergence) across 5 chains and produced effective samples sizes between 50 and 70 for all parameters (approximately 8-10% of the dependent sample size). The values are reported in Table 2.

Following assessment of posterior convergence, the means of each of the posterior parameter estimates were computed and the fitted distribution based on our mixture model (Eq. 14) and estimated parameter values (Table 2) were plotted

over a histogram of the dataset. In order to assess the fit of the estimated distribution to the distribution of the data, a non-parametric method for estimating a distribution known as the Kernel Density Estimate (KDE) was computed for the data and considered the target or 'true' distribution. The KL divergence was then computed between the fitted distribution (with estimated parameters from Gibbs sampling) and the KDE distribution, resulting in a KL divergence of 0.2746, indicating a good fit to the data (Fig. 14). Additionally, the data segmentation into growing, paused, and shrinking states obtained by the Gibbs sampler is shown in Figure 15, indicating a clear segmentation of velocity data into distinguishable growing, paused, and shrinking states with the hypothesized log-N (for growing), Gaussian (for paused), and negative log-N (for shrinking) velocity distributions.

5. CONCLUSION

The results indicate that the Gibbs sampling algorithm can successfully be applied to parameterize mixture models of dendrite branch velocities. However, it is important to note that initialization appears to play an important role in the success of the Gibbs sampler for this case. Using Otsu's method allows for initiating the sampler closer to the true posteriors, allowing the sampler to successfully converge to the true posterior. Further investigation into initialization and the shortcomings of Gibbs sampling algorithms for mixture models and multimodal posteriors may be necessary.

The good fit of our distribution to the data (Fig. 14) and the reasonable segmentation (Fig. 15) further indicates that our choice of a 3-component log-N-Gauss-log-N mixture model accurately models the data. This supports the Howard lab's hypothesis that neuronal Class IV dendrites do indeed display distinguishable growing, paused, and shrinking states as can be observed in microtubules, supporting the hypothesis that dendrite dynamics are driven by microtubule dynamic instability. These results may provide further insight into the underlying biological mechanisms behind dendrite morphogenesis.

The results additionally provide a more rigorous means of quantifying model parameters with interpretable confidence intervals as well as a rigorous method for segmenting experimental data into proposed states with an associated probability. This can improve methods for modelling and simulating dendrite morphogenesis, improving our mechanistic and systems-level understanding of neural development. Furthermore, future studies may reveal differences in model parameters between wild-type (or healthy) neurons and mutant (or diseased-state) neurons which may be used to explain observable differences in dendrite branching patterns, providing a dendrite morphology-based explanation for the emergence of neurological disease.

Since healthy cognitive functioning as well as certain neurological diseases have been linked to dendrite development, the results of this study and future studies on mutant dendrites may in the long-term help provide more insight into the importance of dendrite dynamics in proper neural development and how deviations in dendrite dynamics may contribute to the emergence of neurological disease.

FIGURES

Figure 1: Raw Dendrite Branch Velocity Histogram

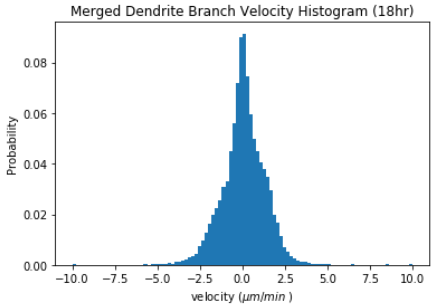


Figure 2: A simulation showing the dwell time of the ion channel in any given state distributed exponentially

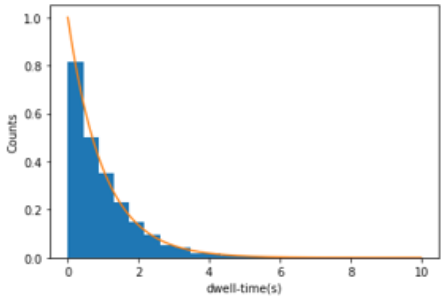
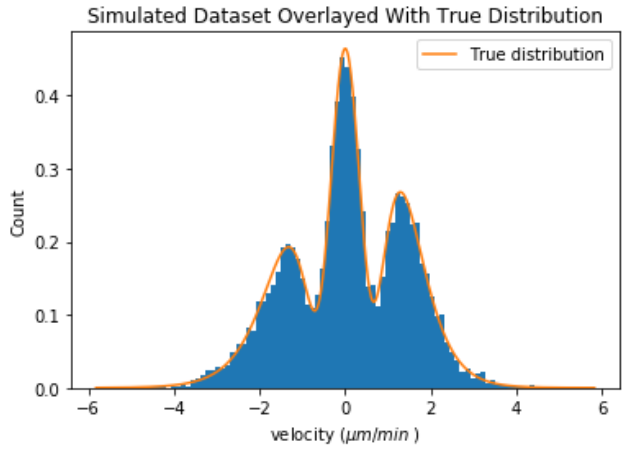


Figure 3: Simulated Dendrite Branch Velocities



A simulated dataset (blue histogram) of dendrite branch velocities according to a 3-component log-N-Gauss-log-N mixture Model (shown as orange distribution). True parameter values were set to values previously obtained by the Howard lab using least squares fitting to fit a log-N-Gauss-log-N mixture model to dendrite branch velocity data. True parameter values were set as: [$true\_\mu_{growing} = 0.3873$, $true\_\mu_{paused} = 0$, $true\_\mu_{shrinking} = 0.4369$, $true\_\sigma_{growing} = 0.3624$, $true\_\sigma_{paused} = 0.3387$, $true\_\sigma_{shrinking} = 0.3918$, $true\_w_{growing} = 0.3351$, $true\_w_{paused} = 0.3939$, $true\_w_{shrinking} = 0.271$]

Figure 4: MCMC chain and posterior for μ parameter using simulated dataset and random initialization. Red line represents true parameter value.

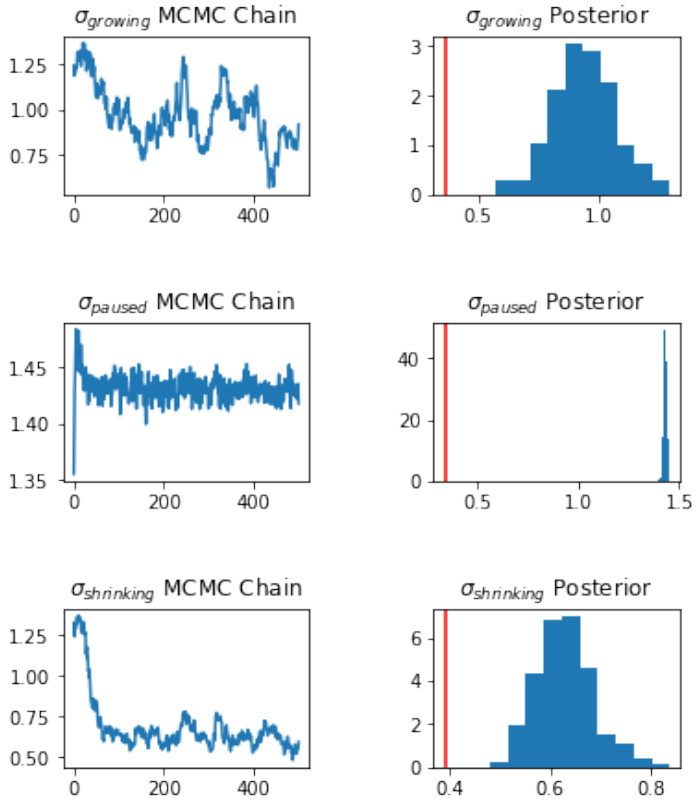


Figure 5: True model distribution (shown in blue) overlayed with the distribution obtained by Gibbs sampling (shown in orange). Parameter estimates were obtained by taking the mean of the posteriors obtained by Gibbs sampling.

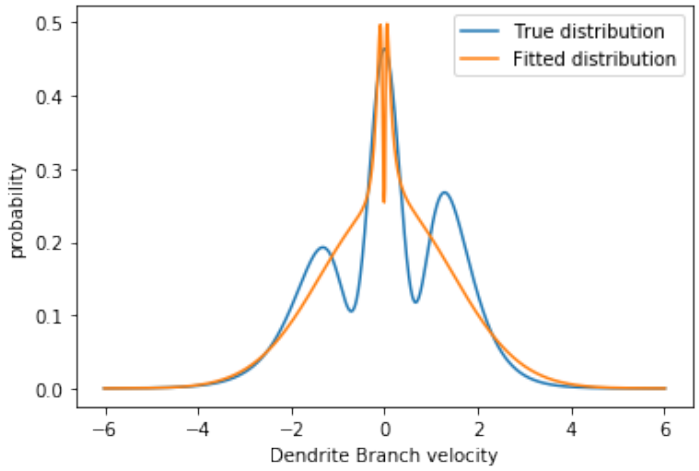


Figure 6: Simulated dataset thresholded into 3 categories using Otsu's method. Thresholds are k1 = -0.919 and k2 = 0.714

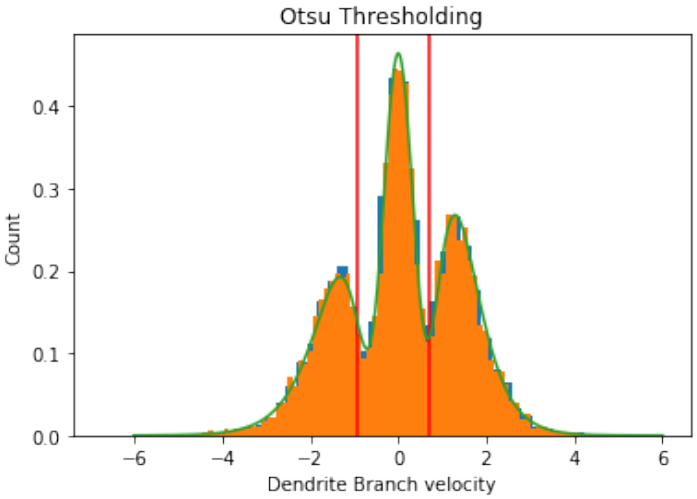


Figure 7: MCMC chain and posterior for μ parameter using simulated dataset and Otsu initialization. μ_{paused} parameter fixed to 0. Red line represents true parameter values. Green lines represent 95% confidence intervals (values shown in Table 1). MCMC chain run for 1000 iterations.

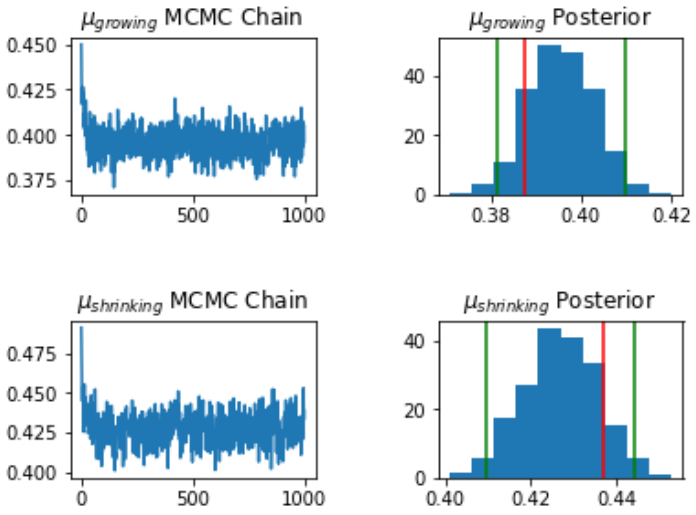


Figure 8: MCMC chain and posterior for σ parameter using simulated dataset and Otsu initialization. Red line represents true parameter values. Green lines represent 95% confidence intervals (values shown in Table 1). MCMC chain run for 1000 iterations.

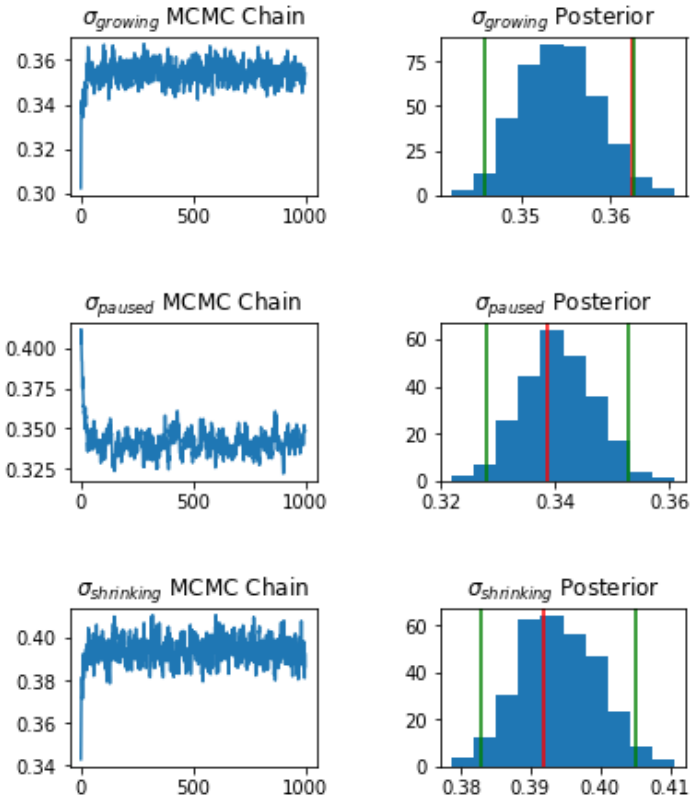


Figure 9: MCMC chain and posterior for w parameter using simulated dataset and Otsu initialization. Red line represents true parameter values. Green lines represent 95% confidence intervals (values shown in Table 1). MCMC chain run for 1000 iterations.

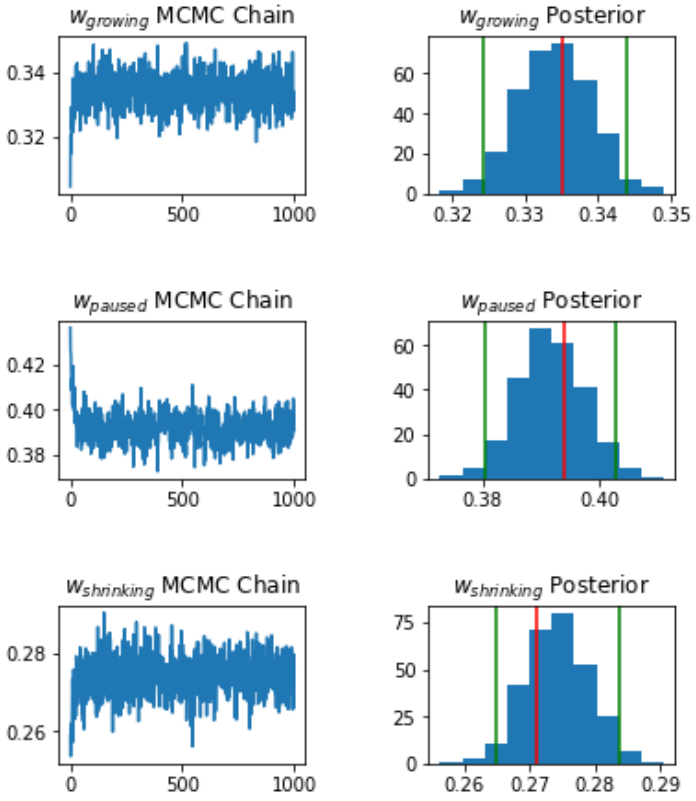


Figure 10: The true model distribution (shown in blue) overlayed with the distribution obtained by Gibbs sampling (shown in orange). Gibbs sampling with Otsu's initialization successfully recovers the true distribution with a KL divergence of 0.0195.

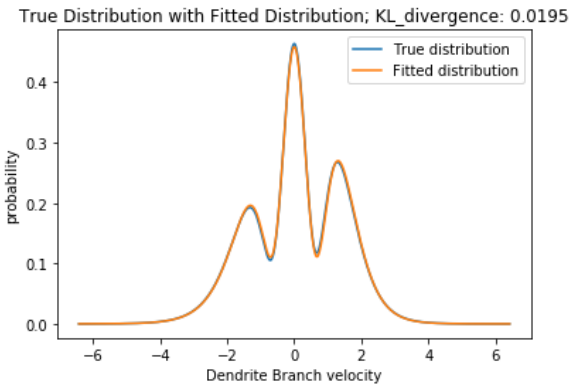


Figure 11: MCMC chain and posterior for μ parameter using experimentally measured dendrite branch velocity dataset and Otsu initialization. MCMC chain was run for 1000 iterations with effective sample sizes shown in Table 2. Red lines represent 95% confidence intervals (values shown in Table 2).

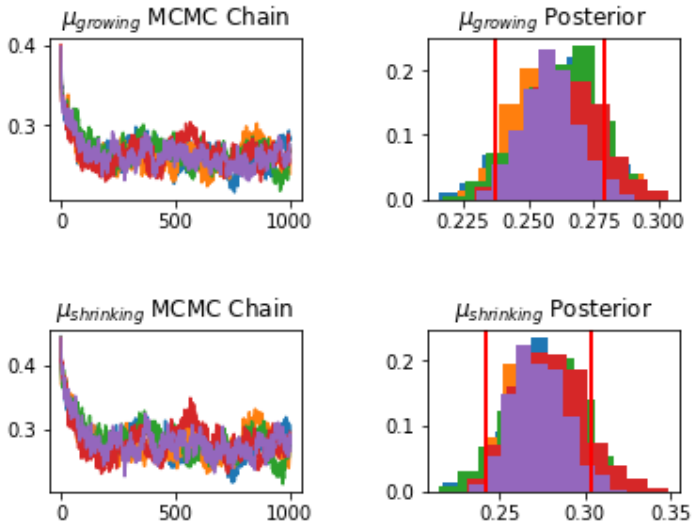


Figure 12: MCMC chain and posterior for σ parameter using experimentally measured dendrite branch velocity dataset and Otsu initialization. MCMC chain was run for 1000 iterations with effective sample sizes shown in Table 2. Red lines represent 95% confidence intervals (values shown in Table 2).

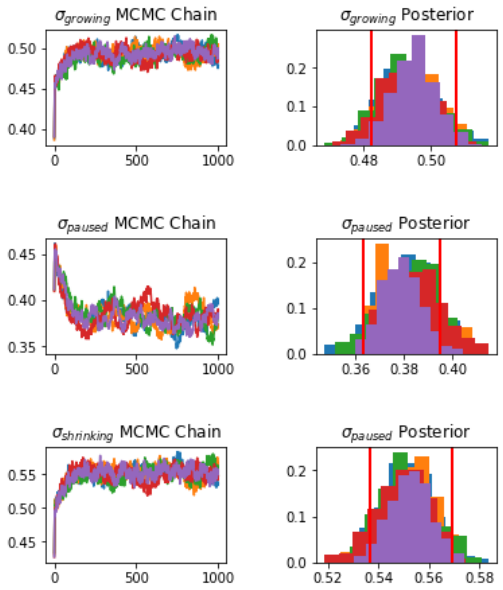


Figure 13: MCMC chain and posterior for w parameter using experimentally measured dendrite branch velocity dataset and Otsu initialization. MCMC chain was run for 1000 iterations with effective sample sizes shown in Table 2. Red lines represent 95% confidence intervals (values shown in Table 2).

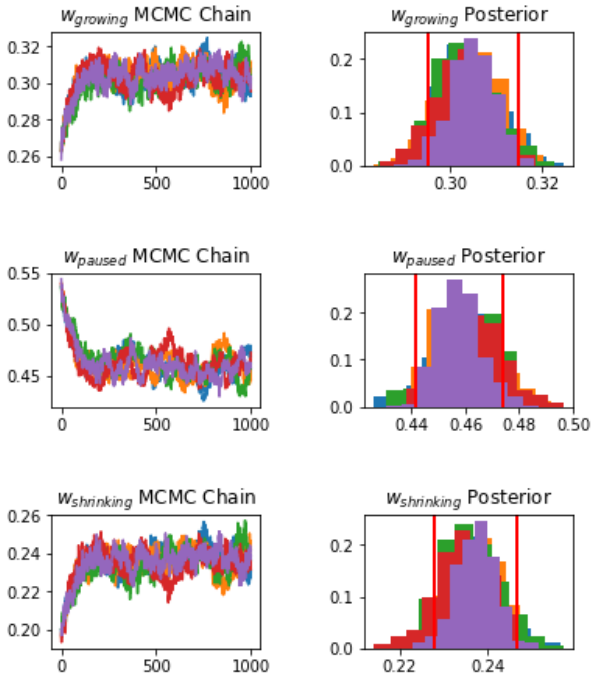


Figure 14: The dendrite branch velocity histogram (shown in yellow) with the KDE distribution (shown in red) overlayed with the distribution obtained by Gibbs sampling (shown in blue). Gibbs sampling with Otsu's initialization recovers parameters that result in a good fit to the data with a KL divergence of 0.2746.

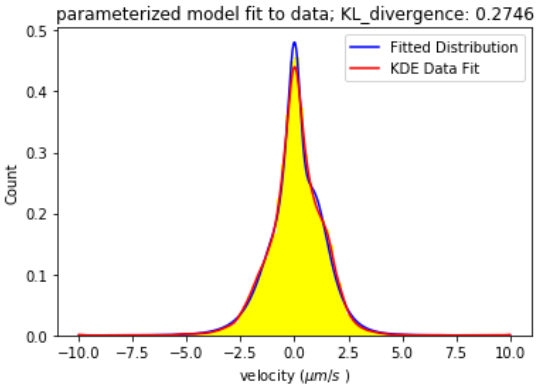


Figure 15: Results of final Gibbs sampling data segmentation into growing, paused, and shrinking states.

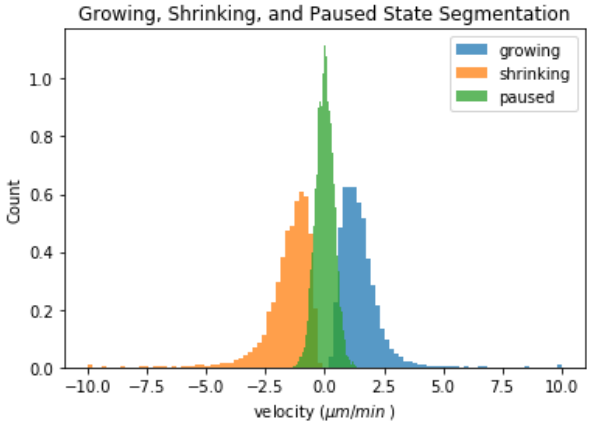


Table 1: Gibbs sampling parameterization for simulated model with known true parameters. Mean of Gibbs sampling estimated posteriors and 95% confidence intervals shown.

Parameter	True Parameter Value	Mean Gibbs Sampling Parameter Value	95% Confidence Interval
$\mu_{growing}$	0.3873	0.3956	(0.3814, 0.4098)
μ_{paused}	0	Fixed at 0	N/A
$\mu_{shrinking}$	0.4369	0.4269	(0.4095, 0.4443)
$\sigma_{growing}$	0.3624	0.3543	(0.3459, 0.3627)
$\sigma_{pauseed}$	0.3387	0.3406	(0.3282, 0.353)
$\sigma_{shrinking}$	0.3918	0.3940	(0.3828, 0.4052)
$w_{growing}$	0.3351	0.3341	(0.3243, 0.3439)
w_{paused}	0.3939	0.3916	(0.3803, 0.403)
$w_{shrinking}$	0.271	0.2742	(0.2647, 0.2838)

Table 2: Gibbs sampling parameterization of Class IV dendrite branch velocity data. Mean parameter values for Gibbs sampling posterior estimates across 5 MCMC chains shown along with 95% confidence intervals. Gelman-Rubin diagnostic shown to assess MCMC chain convergence with a convergence threshold of 1.1. Effective sample size shown for all parameters.

Parameter	Mean Parameter Value (across 5 chains)	95% Confidence Interval (for 1 chain)	Gelman-Rubin Convergence diagnostic (r_hat < 1.1)	Effective Sample Size (dependent sample size = 700)
$\mu_{growing}$	0.2609	(0.2371, 0.2794)	1.0787	63.95
μ_{paused}	Fixed at 0	N/A	N/A	N/A
$\mu_{shrinking}$	0.2760	(0.2424, 0.3035)	1.0788	61.55
$\sigma_{growing}$	0.4936	(0.4823, 0.5075)	1.0769	69.81
$\sigma_{pauseed}$	0.3816	(0.3632, 0.3951)	1.0956	52.01
$\sigma_{shrinking}$	0.5514	(0.5365, 0.5689)	1.0720	68.66
$w_{growing}$	0.3039	(0.2953, 0.3148)	1.0765	67.34
w_{paused}	0.4600	(0.4414, 0.4741)	1.0921	55.67
$w_{shrinking}$	0.2361	(0.2279, 0.2466)	1.0822	64.47

ACKNOWLEDGEMENTS

I would like to thank Prof. Jonathon Howard, Olivier Trottier, Sabya Sutradhar and the rest of the Howard lab for their support and feedback throughout this project.

REFERENCES

(1) Jan, Lily et al., 2010. Branching Out: Mechanisms of Dendritic Arborization. *Nat Rev Neurosci*, Vol 11 pp. 316-328.

(2) Conde, C., Cáceres, A., 2009. Microtubule assembly, organization and dynamics in axons and dendrites. *Nat Rev Neurosci,* Vol 10 pp. 319–332.

(3) Lambert, Ben. A Students Guide to Bayesian Statistics. SAGE, 2018.

(4) Barker, B.S. et al., 2017. Conn's Translational Neuroscience: Chapter 2 - Ion Channels. *Academic Press*. pp. 11-43.

(5) Fridman, Daniel. Bayesian Inference for the Parameterization of Mixture Models with Biophysical Applications, S&DS 480 Independent Study Report, Yale University. December 2019.

(6) Hines, K., 2015. A Primer on Bayesian Inference for Biophysical Systems. *Biophysical Journal*, Vol 108 pp. 2103-2113.

(7) Jordan, M. The Conjugate Prior for the Normal Distribution, lecture notes, Stat260: Bayesian Modeling and Inference, UC Berkeley, delivered February 8 2010.

(8) Otsu, Nobuyuki, 1979. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions and Systems*, Vol SMC-9 pp. 62-66.

(9) Gelman, A. Rubin, D.B., 1992. Inference from Iterative Simulation Using Multiple Sequences. *Statist. Sci*., Vol 7 pp. 457-511.

(10) Kullback, S., Leibler, R.A., 1951. On Information and Sufficieny. *Annals of Mathematical Statistics*. Vol 22 pp. 70-86.

APPENDIX

Conjugate Priors

In certain cases, an exact closed-form solution for the posterior can be calculated without having to calculate the marginal posterior by selecting a mathematically convenient prior. More specifically, if a prior is chosen from a specified family of distributions such that the posterior will fall into the same family of distributions, it may be possible to obtain a closed-form solution for the posterior. These 'mathematically convenient' priors are known as *conjugate priors* (2).

In order to explain how conjugate priors work, it is easiest to use an example. Thus, I will use a biophysically relevant example relating to ion channel patch-clamp recordings in order to demonstrate the use of conjugate priors (5,6).

Example: Ion Channel Patch-Clamp Recording (4,5,6)

Most cells, including neurons, contain proteins called ion channels on their membranes which allow for ions to flow between the interior and exterior of the cell. These ion channels regulate the concentration of ions across the membrane by stochastically transitioning between open and closed states according to a Poisson process. The time an ion channel spends in any given state (dwell-time) is known to follow an exponential distribution. An experiment can be carried out which tracks the time spent in each state and a histogram of dwell-times can be plotted, a simulation of which is shown in Fig. 2.

We first model the dwell-times as random variables from an exponential distribution, $y_i \sim \lambda e^{-\lambda y}$. For N samples, we thus form an exponential likeihood: $\prod_{i=1}^{N} \lambda\, e^{(-\lambda y_i)}$. Next, we seek to determine the time-scale parameter λ of our model based on our data. We can formulate this problem in terms of Bayesian inference as follows:

$$p(\lambda|y) \propto (\prod_{i=1}^{N} \lambda\, e^{(-\lambda y_i)}) p(\lambda)$$

Our goal is to select an appropriate prior, $p(\lambda)$, such that we can obtain a closed form posterior for the time-scale parameter, $p(\lambda|y)$. The conjugate prior to an exponential likelihood is the Gamma distribution:

$$p(\lambda) \sim Gamma(\lambda|\alpha,\beta) = \frac{\beta^{\alpha}}{\Gamma(a)}\lambda^{\alpha-1}e^{-\lambda b}$$

With the following set of steps we can see how the Gamma prior conveniently combines with the exponential likelihood:

$$\begin{aligned} p(\lambda|y) &\propto (\prod_{i=1}^{N} \lambda\, e^{(-\lambda y_i)}) \times \frac{\beta^{\alpha}}{\Gamma(a)}\lambda^{\alpha-1}e^{-\lambda\beta} \\ &\propto (\lambda^{N} e^{-\lambda \sum_{i=1}^{N} y_i}) \times (\frac{\beta^{\alpha}}{\Gamma(a)}\lambda^{\alpha-1}e^{-\lambda\beta}) \\ &\propto (\lambda^{N} e^{-\lambda \sum_{i=1}^{N} y_i})(\lambda^{\alpha-1}e^{-\lambda\beta}) \\ &= \lambda^{\alpha+N-1} e^{-\lambda(\sum_{i=1}^{N} y_i+\beta)} \end{aligned}$$

We observe that the simplified solution above follows the same form as the Gamma distribution, but with new hyperparameters, updated according to our data. Thus, we obtain the final closed-form solution for our posterior:

$$p(\lambda|y) \sim \Gamma(\lambda|\alpha',\beta') \text{ s.t. } \alpha' = \alpha + N \text{ and } \beta' = \sum_{i=1}^{N} y_i + \beta$$

Using the idea of conjugate priors, we are able to solve for the posterior distribution of the time-scale parameter for our exponential model, dependent on our data of ion channel dwell-times.